Recomendação de Jogos usando Filtragem Colaborativa e Baseada em Conteúdo

Rafael Poddis Busquim e Silva

Instituto de Ciência e Tecnologia

Universidade Federal de São Paulo

São José dos Campos, São Paulo

poddis.rafael@unifesp.br

I. Introdução

Com o crescente número de dados na internet ao longo do tempo, foi ficando cada vez mais díficil para o usuário escolher o que consumir, por causa disso, foram surgindo novas técnicas para ajudar o consumidor. Sistemas de recomendação estão sendo bastante utilizados por grandes empresas nos dias atuais, como as plataformas de *streaming*, redes sociais, plataformas de jogos, entre outros. No geral, as recomendações são geradas a partir das caraterísticas do produto consumido pelo usuário contribuindo para uma experiência mais satisfatória, fornecendo ao usuário conteúdo relevante e de interesse. O sistema de recomendação abordado neste trabalho utiliza duas técnicas mais comuns, sendo a filtragem colaborativa e a filtragem baseada em conteúdo.

II. Conceitos Fundamentais

A. Filtragem Colaborativa

Imagina que você tem um grupo de amigos com gostos musicais semelhantes. Quando você quer descobrir uma nova música, você pergunta a eles o que eles estão ouvindo. Eles te recomendam algumas músicas e você acaba gostando de algumas delas. Isso é basicamente como funciona a filtragem colaborativa. É um sistema que usa as preferências de um grupo de usuários para fazer recomendações personalizadas para cada usuário individual. O funcionamento é bem simples, primeiro é feita a coleta de informações sobre as prefrências de usuários. Depois o sistema procura por usuários que têm gostos parecidos com os seus, por exemplo, se você e outras pessoas deram uma boa avaliação para os mesmos jogos e por fim o sistema irá recomendar conteúdo que você ainda não viu e que provavelmente vai gostar, com base nas preferências dos usuário semelhantes.

B. Filtragem Baseada em Conteúdo

A filtragem baseada em conteúdo é uma técnica que recomenda itens, como jogos, que um usuário possa gostar, com base nas características desses itens e nas preferências do próprio usuário. Imagine que você está em uma livraria e você sabe que gosta de livros de ficção científica e mistério. Com base nessa informação, o vendedor da livraria recomenda outros livros de ficção científica e mistério, porque sabe que você provavelmente vai gostar desses tipos de livros. Assim como na filtragem colaborativa, a execução dessa técnica é facil, o sistema começa coletando informações sobre os itens e suas características, por exemplo, para jogos, pode ser coletado informações como o gênero, se é um jogo de aventura ou estratégia. Em seguida, é criado um perfil de usuário com base nos itens que ele gostou anteriormente e finalmente ele encontra itens semelhantes ao que o usuário gostou.

C. Algortimo

O algoritmo usado neste projeto é chamado k-Nearest Neighbors (kNN), que é usado tanto para problemas de classificação quanto de regressão. O *k* é uma constante de sua escolha, por exemplo, 5, então o algoritmo vai selecionar uma amostra aleatória nos dados e analisar as 5 amostras mais próximas da inicial. Se a maioria dessas 5 amostras forem do tipo A então a primeira amostra escolhida também é do tipo A, por outro lado, se a maioria for do tipo B, então a primeira também é do tipo B. As grandes

desvantagens desse algoritmo é que pode ser computacionalmente custoso para grandes conjuntos de dados, pois precisa calcular a distância para todos os dados de treino e a escolha do *k* pode influenciar significamente os resultados. Por exemplo, é preferível não escolher um número par, pois poderia não haver uma maioria entre as amostras. Se *k=4*, teria possibilidade de terem 2 amostras tipo A e 2 tipo B, o que dificultaria a decisão do algoritmo.

III. Trabalhos Relacionados

Diversos trabalhos exploram sistema de recomendação, com o uso de filtragem colaborativa ou baseada em conteúdo ou até mesmo um sistema híbrido, envolvendo as duas técnicas [1]. Outros, utilizam técnicas diferentes do modelo kNN utilizado neste trabalho. O trabalho de J. CAO, Z. WU, Y. ZHUANG, B. MAO e Z. YU [2], usa métodos de kernel na tentativa de melhorar a precisão das previsões e também percebeu que ao usar o modelo kNN, problemas como uso elevado de memória podem surgir.

Um artigo propõe um sistema usando filtragem colaborativa utilizando inferência variacional [3], que aborda desafios como o problema de KL-vanishing em autoencoders variacionais, visando aumentar a robustez e a precisão das recomendações. Em outro estudo, Q. -H. Le, T. N. Mau, R. Tansuchat and V. -N. Huynh [4] utilizaram um método de filtragem colaborativa multi-critério que combina deep learning (aprendizagem profunda) com a teoria de Dempster-Shafer.

IV. Metodologia Experimental

Para a realização desse trabalho, foi utilizado dados de jogos e usuários fornecidos pelo Kaggle. No total eram três arquivos do tipo *.csv*, games.csv que contém os nomes, preços, plataformas jogadas, entre outras colunas a respeito dos jogos, recommendations.csv que representa mais de 41 milhões de relações entre jogos e usuários, mas para não deixar o processamento lento, foi reduzido para aproximadamente 210 mil dados e users.csv que contém o perfil de cada usuário, número de jogos jogados e avaliações publicadas, porém essa base não é relevante para este trabalho então não foi utilizada. Também um arquivo *.json* que contém a descrição e *tags* de um jogo, que é uma lista de gêneros em que determinado jogo se encaixa.

Após o carregamento dos dados, alguns gráficos foram gerados para termos uma ideia melhor dos dados que estamos utilizando. O primeiro gráfico é um histograma que apresenta a distribuição dos valores discretos, se um jogo foi recomendado ou não. Há também um Boxplot para identificar possíveis outliers significativos na distribuição de horas jogadas, caso haja, podemos tratá-los para melhorar a qualidade das recomendações. E por fim, foi gerado uma matriz de correlação, que mostra a correlação entre pares de variáveis como as colunas 'hours' e 'is_recommended'.

As bases contém colunas que não serão utilizadas no sistema de recomendção, por isso, é necessário tirá-las. Na base games.csv, foram utilizadas apenas cinco colunas, app_id, title, rating, positive_ratio, user_reviews e price_final. Já na base recommendations.csv, foram usadas apenas quatro colunas, app_id, user_id, hours e is_recommended. Foram verificados também se haviam valores nulos nessas bases, porém não tinha nenhum valor faltante. Por outro lado, no arquivo, games_metadata.json, foram encontrados, não valores nulos, mas sim descrições em branco e listas de *tags* vazias, então foi feito um tratamento para retirar esses dados.

Um dos grandes desfios neste trabalho foi sem dúvidas a grande utilização de memória na hora de gerar a matriz de utilidade. A matriz de utilidade é feita a partir de três parâmetros, os valores, que foram gerados pela multiplicação das colunas de horas jogadas e a coluna que diz se foi recomendado ou não, pois se um jogador jogou muitas horas, porém não recomendou o jogo, então não é relevante. Também na matriz temos as colunas, que foram compostas pelos ID's dos jogos e as linhas compostas pelos ID's dos usuários. Essa matriz demorava muito para ser gerada ou usava toda a memória disponível e reiniciava o processamento, mesmo reduzindo os dados e tentando outra técnicas para reduzir o uso de memória, na hora de treinar o modelo acontecia o mesmo problema. Então foi descoberto que a matriz de utilidade estava sendo gerada de forma errada e para ajustar isso, foi preciso normilizar os ID's, para que começassem do zero ao invés de um número aleatório.

A base recommendations.csv foi dividida em treino e teste utilizando o método *"train_test_split"* da biblioteca Scikit-learn, sendo 80% da base para treino e os outros 20% para teste. Foi criada a matriz de utilidade a partir do conjunto de treino e a matriz de similariidade de cosseno com a matriz de utilidade. Logo após esse processo, treinamos o modelo kNN visto anteriormente neste trabalho. O modelo recebe três parâmetros, número de vizinhos, algoritmo e métrica. O primeiro parâmetro define quantos vizinhos próximos o modelo terá que comparar e para computar esses vizinhos próximos ele precisa de um algoritmo que foi definido como

‘auto’, ou seja, irá decidir qual o algoritmo mais apropriado com base nos valores passados no treino. Por fim, a metrica utilizada foi *‘cosine’*, já que foi feita uma matriz de similaridade usando cosseno.

Como protocolo de validação do modelo, será utilizado algumas métricas como Precision e Recall para avaliar a qualidade das recomendações. Uma matriz de confusão para visualizar os erros e acertos. Por último será feita uma validação cruzada para ajustar e melhorar a estabilidade do modelo.

V. Conclusões

VI. Referências

[1] R. G. N. Silva, "Sistema de Recomendação baseado em conteúdo textual: avaliação e comparação," M.S. thesis, Univ. Federal da Bahia, Univ. Estadual de Feira de Santana, Salvador, BA, Brazil, 2023.

[2] J. CAO, Z. WU, Y. ZHUANG, B. MAO and Z. YU, "A Novel Collaborative Filtering Using Kernel Methods for Recommender Systems," in Chinese Journal of Electronics, vol. 21, no. 4, pp. 609-614, October 2012.

[3] K. Zheng, X. Yang, Y. Wang, Y. Wu and X. Zheng, "Collaborative filtering recommendation algorithm based on variational inference," in International Journal of Crowd Science, vol. 4, no. 1, pp. 31-44, March 2020.

[4] Q. -H. Le, T. N. Mau, R. Tansuchat and V. -N. Huynh, "A Multi-Criteria Collaborative Filtering Approach Using Deep Learning and Dempster-Shafer Theory for Hotel Recommendations," in IEEE Access, vol. 10, pp. 37281-37293, 2022.